ID based cryptography for secure cloud data storage

Nesrine Kaaniche, Aymen Boudguiga, Maryline Laurent

ID-Based Cryptography for Secure Cloud Data Storage

Nesrine Kaaniche[1], Aymen Boudguiga[2], Maryline Laurent[1]
[1]Institut Mines-Telecom, Telecom SudParis, UMR CNRS 5157 SAMOVAR
9 rue Charles Fourier, 91011 Evry, France
e–mail: {Nesrine.Kaaniche, Maryline.Laurent}@telecom-sudparis.eu
[2]CEA, LIST, Communicating System Laboratory
Saclay, 91400, Gif-sur-Yvette, France
e–mail: Aymen.Boudguiga@cea.fr

***Abstract*—This paper addresses the security issues of storing sensitive data in a cloud storage service and the need for users to trust the commercial cloud providers. It proposes a cryptographic scheme for cloud storage, based on an original usage of ID-Based Cryptography. Our solution has several advantages. First, it provides secrecy for encrypted data which are stored in public servers. Second, it offers controlled data access and sharing among users, so that unauthorized users or untrusted servers cannot access or search over data without client's authorization.**

I. Introduction

Nowadays, the use of cloud based services for large scale is gaining an expanding interest. The National Institute of Standards and Technology (NIST) [1] defines the cloud computing as a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources. These resources can be storage capacities that are controlled, allocated and managed by the Cloud Service Provider (CSP). Therefore, by moving their data to the cloud, users remove the burden of building and maintaining a local storage infrastructure. As such, they only have to pay their CSP for the allocated resources. Microsoft Windows Azure storage services [2] and Amazon's Simple Storage Service (S3) [3] are good examples. Indeed, these providers offer to their clients the possibility to store, retrieve and share data with other users in a transparent way.

Unfortunately, in addition to its advantages, cloud storage brings several security issues. Data confidentiality appears as the biggest concern for users of a cloud storage system. In fact, the clients' data are managed out of their governance. Kamara and Lauter [4], and Chow et al. [5] agreed that encrypting outsourced data by the client is a good alternative to mitigate such concerns of data confidentiality. Thus, the client preserves the decrypting keys out of reach of the cloud provider. The confidentiality provisioning becomes more complex with flexible data sharing among a group of users. It requires efficient sharing of decrypting keys between different authorized users. So that, only authorized users are able to obtain the cleartext of data stored in the cloud.

In this paper, we describe a new method for improving data confidentiality in cloud storage systems and enhancing dynamic sharing between users. It can be used by an authenticated client for his data storage, backup and sharing in the cloud. Our proposal relies on the use of ID-Based Cryptography (IBC), where each client acts as a Public Key Generator (PKG). That is, he generates his own public elements and derives his corresponding private key using a secret. IBC is yet another method to generate public and private keys [6]. It considers the client identity as his public key, and derives a corresponding private key using a secret.

The originality of our proposal is twofold. First, it ensures better confidentiality. That is, every client acts as a PKG by computing an ID-based pair of keys to encrypt the data that he intends to store in the cloud. As such, the data access is managed by the data owner. Second, by using a per data ID-based key, we provide a flexible sharing approach. Indeed, the distribution of decrypting keys between the client and the authorized users, does not reveal any information about the client's secret.

The remainder of this work is organized as follows. First, we describe, in Section II, the cloud architecture considered in our work. Then, we introduce the ID-Based Cryptography in Section III and some of the related works in Section IV. Next, we describe in Section V our contribution. Section VI gives a security analysis of our proposal. Finally, we discuss the evaluation results in Section VII, before concluding in Section VIII.

II. Cloud Storage

We present, in this section, a typical cloud storage architecture. Then, we review its related security threats and requirements.

A. Architecture

Figure 1 illustrates a descriptive network architecture for cloud storage. It relies on the following entities for the good management of a client data:

- *Cloud Service Provider (CSP)*: a CSP has significant resources to govern distributed cloud storage servers and to manage its database servers. It also provides virtual infrastructure to host application services. These services can be used by the client to manage his data stored in the cloud servers.
- *Client*: a client makes use of provider's resources to store, retrieve and share data with multiple users. A client can be either an individual or an enterprise.
- *Users*: the users are able to access the content stored in the cloud, depending on their access rights which are authorizations granted by the client, like the rights to read, write or re-store the modified data in the cloud. These access rights serve to specify several groups of users. Each group is characterized by an identifier ID_G and a set of access rights.

In practice, the CSP provides a web interface for the client to store data into a set of cloud servers, which are running in a cooperated and distributed manner. In addition, the web interface is used by the users to retrieve, modify and re-store data from the cloud, depending on their access rights. Moreover, the CSP relies on database servers to map clients identities to their stored data identifiers and groups identifiers.

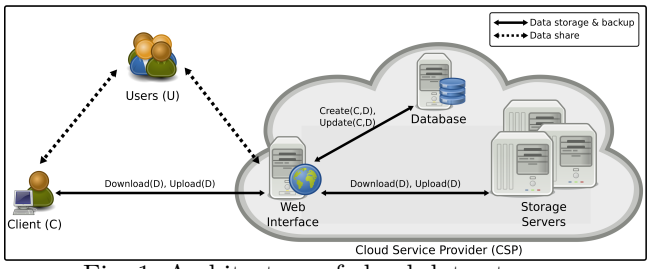

Fig. 1: Architecture of cloud data storage

B. Security Requirements

When outsourcing data to a third party, providing confidentiality and privacy becomes more challenging and conflicting. Privacy is a critical concern with regards to cloud storage due to the fact that clients' data reside among distributed public servers. Therefore, there are potential risks where the confidential information (e.g., financial data, health record) or personal information (e.g., personal profile) are disclosed. Meanwhile, confidentiality implies that client's data have to be kept secret from both cloud provider and other users. Confidentiality remains as one of the greatest concerns. This is largely due to the fact that users outsource their data on cloud servers, which are controlled and managed by potentially untrusted CSPs. That is why, it is compulsory to provide secrecy by encrypting data before their storage in cloud servers while keeping the decryption keys out of the reach of CSP and any malicious user.

For designing the most suitable security solutions for cloud storage, we are considering an *honest but curious* cloud provider, as a threat model. That is, it honestly performs the operations defined by our proposed scheme, but it may actively attempt to gain the knowledge of the outsourced data.

In addition, an attacker can be either a revoked user with valid data decryption keys, an unauthorized group member or a group member with limited access rights. Therefore, secure data sharing should support flexible security policies including forward and backward secrecy.

- *Forward secrecy* – this property requires that the confidentiality of previous encrypted data has to be ensured even after the long-term secrets are exposed. For example, a user cannot access stored data before he joins a group.
- *Backward secrecy* – this property means that a compromise of the secret key does not affect the secrecy of future encrypted data. A such, a revoked group member is unable to access data that were outsourced after he leaves the group.

Beyond these security requirements, our proposal aims to achieve several security and system performances. First, the overhead of implemented security mechanisms should be acceptable and lightweight storage cost of the encrypting keys. Second, an efficient and highly scalable key distribution should be designed.

III. ID-Based Cryptography

ID-Based Cryptography (IBC) was initially introduced by Shamir [7] to provide entities with public and private key pairs with no need for certificates and CA deployment. Shamir assumes that each entity uses one of its identifiers as its public key. These identifiers have to be unique. In addition, he assigns the private key generation function to a special entity called the Public Key Generator (PKG). That is, before accessing the network, every entity has to contact the PKG to get its private key. This private key is computed so as to be bound to the public key of the entity. During the last decade, IBC has been enhanced by the use of the Elliptic Curve Cryptography (ECC) [8]. As a consequence, new ID-based encryption and signature schemes emerged.

In order to be able to derive a client's private key, the PKG must first define a set of *ID-based public elements* (IBC–PE). The PKG generates the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing function $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_2$ in $\mathbb{G}_T$. $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive subgroups of the group of points of an Elliptic Curve (EC). However, $\mathbb{G}_T$ is a multiplicative subgroup of a finite field. $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ have the same order q. In addition, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are generated by P, Q and the generator $g = \hat{e}(P, Q)$, respectively. The bilinear function $\hat{e}$ is derived from the Weil or Tate pairing ([9]).

After the specification of the groups, the PKG defines a set of hash functions in accordance to the ID-based encryption and signature schemes in use [6]. As such, the PKG defines a hash function $Hash_{pub}()$ to transform the client's identity (ID) into a public key as follows:

$$Pub_{ID} = Hash_{pub}(ID)$$

Generally, the public key of a client is computed as a hash of one of his identities and it is either a point of an elliptic curve [10] or a positive integer [11].

The PKG generates the private key of an entity using a local secret $s_{PKG} \in \mathbb{Z}_q^*$ and a private key generation function $PrivGen()$. Note that the private key is computed as:

$$Priv_{ID} = PrivGen(s_{PKG}, Pub_{ID})$$

For example, Boneh and Franklin [10] compute the private key as $Priv_{ID} = s_{PKG}.Pub_{ID}$, where Pub_{ID} is a point $\in \mathbb{G}_1$. However, Sakai and Kasahara [11] generate the private key as $Priv_{ID} = [1/(Pub_{ID} + s_{PKG})].P$, where Pub_{ID} is an integer. After generating a private key, the PKG has to secure its transmission to its owner either using cryptography or directly to the physical person (using a secure transportation device). The groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the pairing $\hat{e}$, the points P, Q and $Q_{pub} = s_{PKG}.Q$, and the hash functions form the ID-based public elements; IBC–PE=$\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, \hat{e}, g, P, Q, Q_{pub}, Hash_{pub}(), H_1(), ..., H_k()\}$.

IV. Related works

The application of ID-Based Cryptography, in a distributed environment, is an emerging and interesting area, which has been partially investigated in the literature. IBC was first adapted to grid networks. The idea of applying IBC to grid security was explored by Lim and Robshaw in 2004 [12]. In their proposal, each virtual organisation has its own PKG, and all of its users share the same IBC–PE certified by a grid certification authority. Their scheme offers to the encrypting entity more flexibility during the key generation process, and permits to add granularity to the ID-based public key. In fact, Lim and Robshaw propose to include the security policy into the identifier used as input for the public key computation algorithm. However, their proposal has two drawbacks. First, the user needs to maintain an independent secure channel with the PKG for the retrieval of his private key. Second, the PKG is able to achieve a key escrow attack, due to its knowledge of the clients' private keys.

Then, in 2005, Lim and Robshaw [13] introduced a new concept of dynamic key infrastructure for grid, to simplify the key management issues listed in [12]. That is, each user is in charge of publishing his IBC–PE to the other entities. He distributes a fixed parameter set through a X.509 certificate to allow users to act as their own trusted authorities for the purpose of delegation and single sign-on. Therefore, they remove the need for a proxy certification. On one hand, this technique avoids the key escrow attack and the need for a secure channel for private key distribution in an ID-based system. Unfortunately, users have to support the cumbersome task of verifying the parameter sets of other entities. In addition, this paper does not address the arising risk of Man In The Middle attacks [14].

In 2005, Lim and Paterson [15] proposed to use IBC in order to secure a grid environment. They describe several scenarios in which IBC simplifies the current grid solutions, like the elimination of the use of certificate, simple proxy generation, easy revocation of proxy certificates and the savings of bandwidth by using the pairing based approach proposed by Boneh and Franklin [10]. In the same way, Li et al. [16] propose to use IBC as an alternative to the SSL authentication protocol in a cloud environment. However, these schemes still suffer from the needed trust hierarchy to ensure a secure working system.

Recently, Schridde et al. [14] presented a novel security infrastructure, using IBC, for service-oriented cloud applications to overcome the problems of certificate based solutions. In their proposal, the URLs of the service are used for public keys generation.

In Section V, we propose a new approach for a secure cloud storage system, based on IBC.

V. Our proposal for secure data storage, backup and sharing

In this section, we introduce our original idea (Section V-A) before enumerating the prerequisites (Section V-B). Then, we describe in depth our proposed solutions for data storage, backup and sharing.

A. Original idea

Our idea consists in using IBC to provide a per data pair of keys. That is, we propose to use each client as a PKG which generates his own IBC–PE. These IBC–PE are used to compute ID-based keys. These keys serve to encrypt the data before their storage and sharing in the cloud. Note that for every different data, the client computes the corresponding private and public keys relying on his IBC–PE and a local secret s_C.

The choice for IBC is motivated by several reasons. First, we benefit from an easier key management mechanism thanks to the certificate-free feature of IBC. That is, the computation of public keys from the unique data identifiers does not require the deployment of a Public Key Infrastructure (PKI) and the distribution of certificates. Second, IBC permits deriving public keys with no need for previous computation of corresponding private keys. That is, contrary to traditional public key derivation schemes, IBC does not require to generate the private key before the public key. Indeed, users have only to generate ID-based public keys to encrypt data before storage. As such, any user can directly encipher data for a client at no extra cost of communication. The derivation of the corresponding private keys is only needed at the time of data recovery. Third, IBC permits to derive a per data key from a unique data identifier thanks to the lightweight key computation. The derivation of a per data key is well suited for a sharing process. That is, the client uses a different ID-based pair of keys for each new data storage. Therefore, he has merely to reveal the ID-based private key needed for shared data decryption. As such, we avoid the use of the same key for enciphering all the outsourced data. That is, when the private key used for the decryption is captured by an

attacker, he cannot get any information about the other per data keys.

B. Prerequisites

This section gives the prerequisites which we have used for designing our solution. First, we assume that there is an established secure channel between the client and the CSP. This secure channel supports mutual authentication and data confidentiality and integrity. It can be implemented through the Transport Layer protocol (TLS) [17], where the client can authenticate with a certificate or password. TLS permits data to be transmitted securely.

Second, each client generates his own IBC–PE$_C$ that he intends to use to secure his data storage. Note that the client keeps secret s_C which is needed for IBC–PE generation and private keys derivation. We must also note that, in practice, the client should first select the ID-based encryption scheme which will be used for ciphering messages. Our proposal does not depend on the defined scheme. However, that choice depends on the way the private keys are generated.

After successfully authenticating with the CSP, the client starts the storage process as in Section V-C. Indeed, the client enciphers his data using a per data ID-based public key Pub_D that is derived from the concatenation of the client's identity ID and the data identifier ID_D, as follows:

$$Pub_D = Hash_{pub}(ID_C||ID_D)$$

ID_D is locally generated by the client and is derived from the meta-data (MD) using a one way function $H()$ as $ID_D = H(MD)$. We assume that MD support the data model as specified by the Cloud Data Management Interface standard (CDMI) [18]. Our choice to hide the content of MD is motivated by the need to ensure the data privacy of a client.

The different notations used in this paper are listed in Table I.

Notation	Description
D	data
IBC–PE$_C$	ID-Based Public Elements of the client
ID_C	identity of the client (data owner)
ID_D	data identifier
ID_G	group identifier
ID_U	user identity
MD	meta-data
$Priv_D$	private key associated with the data
Pub_D	public key associated with the data
s_C	secret kept locally the client

TABLE I: Notations used in this paper

C. Secure Data Storage

When a client wants to store data in the cloud, he has to generate the data identifier ID_D. This identifier, associated to a client's identity, must be unique in the CSP database. Thus, the client starts the storage process by sending a *ClientRequestVerif* message to verify the uniqueness of the generated ID_D to his CSP.

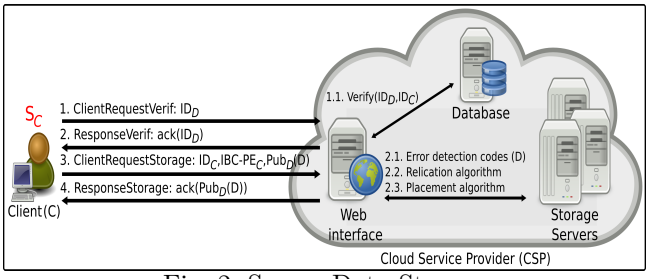

Fig. 2: Secure Data Storage

The storage process consists in exchanging the four following messages (cf. fig 2):

- *ClientRequestVerif*: this first message contains the generated data identifier ID_D. This message is a request for the verification of the uniqueness of the ID_D. More specifically, the client sends the ID_D derived from MD in order to verify the uniqueness of the data identifier in the cloud database servers. The CSP replies with a *ResponseVerif* message to validate or unvalidate the claimed identifier. Note that the data storage process has to be stopped when the uniqueness verification fails and the client has to re-star the storage process and generate a valid data identifier. Therefore, the CSP does not accept a data identifier unless it does not exist in its database.
- *ResponseVerif*: This acknowledgement message is generated by the CSP to validate the requested ID_D. When receiving this message, the client concatenates ID_C and ID_D for deriving the public key Pub_D used to encipher his data.
- *ClientRequestStorage*: it contains the public elements generated by the client IBC–PE$_C$ and the encrypted data Pub_D(D). Note that s_C is kept secret by the client.
- *ResponseStorage*: This acknowledgement message, sent by the CSP, is used to confirm to the client the success of his data storage.

D. Secure Data Backup

The data backup process starts when the client requests for retrieving the data previously stored in the cloud. The data backup process, presented in Figure 3, includes two messages:

- *ClientRequestBackup*: it contains the data identifier ID_D of the requested data that the client wants to retrieve.
- *ResponseBackup*: in this response, the CSP includes the encrypted outsourced data Pub_D(D). Upon receiving the *ResponseBackup* message, the client derives the per data private key $Priv_D$ from the local stored secret s_C and the IBC–PE$_C$, in order to decipher the data.

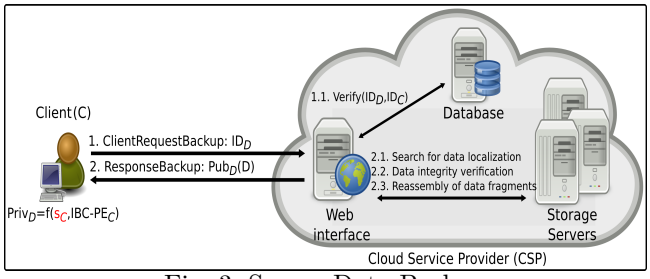

Fig. 3: Secure Data Backup

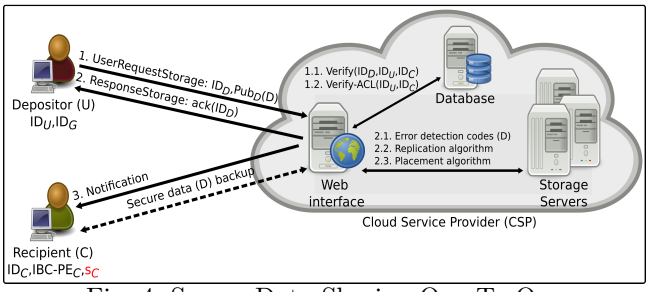

Fig. 4: Secure Data Sharing One To One

E. Secure Data Sharing

We consider the data sharing process, where the client outsources his data to the cloud and authorizes a group of users to access the data. Next, we refer to these user(s) as the recipient(s) and to the data owner as the depositor. Afterwards, we distinguish two different scenarios. First, the data sharing one to one, presented in Section V-E1, where a depositor stores for one recipient. Second, the data sharing one to many, described in Section V-E2, where a depositor shares data among a group of recipients. We must note that our proposal does not require from the recipients to be connected during the sharing process. Indeed, recipients' access rights are granted by the data owner and managed by the CSP. That is, the CSP is in charge of verifying each recipient access permissions before sending him the outsourced data.

1) Scenario E1: Secure Data Sharing One To One: Secure Data Sharing One To One is defined when a depositor wants to share data with one recipient. That is, the depositor ID_U can store encrypted data for this recipient client ID_C by using a per data ID-based public key and the public elements IBC–PE$_C$ of **the recipient**. The depositor will derive the identifier of the data that he intends to share with the recipient and generate the per data public key as follows:

$$Pub_D = Hash_{pub}(ID_U||ID_C||ID_D)$$

This sharing process includes the following messages (cf. fig 4):

- *UserRequestStorage*: this message is a request sent by the depositor that includes the new generated data identifier ID_D and the data encrypted with Pub_D. After verifying uniqueness of ID_D, the CSP stores the data and sends back the *ResponseStorage* message.
- *ResponseStorage*: it is an acknowledgement message sent by the CSP to the requesting depositor. Then, the CSP sends a notification to the recipient to notify the availability of new data enciphered with his public elements. Note that, the CSP also includes, in this notification, the depositor identity ID_U and the data identifier ID_D. When the recipient receives this notification, he starts a backup process, as in Section (V-D).

2) Scenario E2: Secure Data Sharing One To Many: When a depositor wants to share data among a group of recipients, he has first to generate the data identifier ID_D and a selected group identifier ID_G with the access rights granted to the associated users of the group. Then, he computes a per data public, using his **own** public elements as follows:

$$Pub_D = Hash_{pub}(ID_G||ID_D)$$

In practice, each recipient is assumed to know the corresponding private key for decrypting stored data. This private key distribution problem can be solved in two ways. Either the depositor sends the deciphering key to the recipient as soon as he stores data or a proxy is in charge of distributing the private keys. So that, ID-based proxy re-encryption is a suited solution for efficient distribution of secret keys within our cryptographic system [**?**]. That is, the ID-based decrypting keys are encrypted under a public *master* key which is stored on the server side. When a recipient wants to decipher data, the CSP re-encrypts the encapsulated decrypting key from the master key to the key of the requesting recipient. Consequently, the CSP provides access control for encrypted data, but does not possess the ability to decrypt outsourced data on its servers. Once the depositor stored the data with the authorized access rights of the group, each member of the group can start the data sharing process based on the two following messages (cf. fig 5):

- *UserRequestAccess*: This message contains the requested data identifier ID_D. When receiving this message, the CSP searches for the read/write permissions of the recipient, and then, he generates a *Response Access* message.
- *ResponseAccess*: the CSP includes, in its response, the public elements of depositor IBC–PE$_C$ and the encrypted data Pub_D(D).

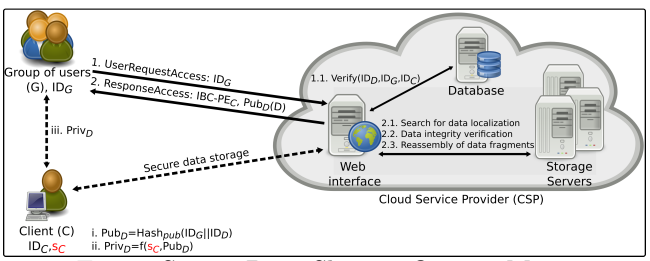

Fig. 5: Secure Data Sharing One To Many

VI. Security discussion

In this section, we give an informal security analysis of our proposal. In addition, we expose its possible refinements to mitigate other threats.

- *Privacy*– Based on cryptographic solution to keep data content secret, sensitive information are generally included in meta-data (e.g., file name, client identity, keywords) [19]. Therefore, in our proposal, we present an ID-based cryptographic solution based on hashed meta-data. As such, these meta-data form the data identifier, which is used to derive the per data public key. Thus, the client has privacy guarantees on his stored data. First, meta-data content can never be disclosed to the CSP, as he only has access to hashed information $ID_D = H(MD)$. Second, the CSP cannot reveal the content of stored data. In fact, although, he has the data identifier and the public elements of the client IBC–PE$_C$, he does not have the secret s_C needed to derive the private key and to decipher data. Furthermore, searching for stored data, for a backup process, may also endanger the privacy. That is, general retrieval methods are based on keywords search. However, the enforcement of these propositions partially violates privacy protection, since the CSP can guess the content of the stored data based on keywords. In our proposal, we replace the usage of clear keywords by the use of pseudo-random data identifiers which are generated by a hash computation over MD. That is, we do not rely on keywords search during data backup.
  Privacy is also threatened by the *accounting* requirement. That is, general accountability approaches include user profiling, information logging, replay, tracing [20], etc. These operations may not be completed without revealing some private information. Unfortunately, this security conflict between accountability and user privacy has not been solved yet by our approach. That is, our proposal is based on identities. Nevertheless, we may rely on third trusted party, to manage a federation identity mechanism.
  Finally, we note that privacy is tightly related to confidentiality, due to the notion that they both prevent information leakage. Therefore, if data confidentiality is ever violated, privacy will also be violated.
- *Data confidentiality*– In this proposal, we perform an ID-based cryptographic solution to ensure data confidentiality for secure data storage, backup and sharing.
  First, we propose to outsource encrypted data to cloud servers. In our approach, the client is in charge of enciphering his data before their storage in the cloud. He acts as a PKG entity and he is responsible for generating and managing his secrets on his own. Thus, he is the only entity knowing the IBC secret s_C. This secret, which is stored locally at the client's, is needed to derive any deciphering key. Therefore, it is impossible for the CSP or a malicious user to retrieve the deciphering key to decrypt data.
  Second, we propose to use a per data key for enciphering data. This proposal is well suited for the sharing process, as, the client uses a different ID-based pair of keys for each new data storage. He has only to reveal the ID-based private key needed for shared data decryption. As such, we avoid using the same key for enciphering all the outsourced data. In fact, when the private key used for the decryption is captured by an attacker, he cannot get any information about the other per data keys.
- *Access control to data*– The proposed secure sharing scheme is designed to provide forward and backward secrecy of outsourced data. As discussed in Section V-E, our scheme, first, authorizes recipients to have access to data, based on their respective access rights.
  The issue of unauthorized access to data is twofold. First, the issued access rights to the recipients are granted by the depositor and managed by the CSP. In addition, when a recipient wants to access outsourced data, he has first to authenticate with the CSP. That is to say, the access to data has already been strictly controlled by an authentication phase, before the verification of the authorizations granted by the depositor. Therefore, the enforcement of the access control policy is safeguarded.
  Second, even though the CSP or a malicious recipient can gain access to the data, the enforcement of data confidentiality is still guaranteed. In fact, they can only have access to encrypted data or to hashed meta-data. They don't have the needed private key to decipher data.
  However, like for any distribution scenario (e.g., sharing one to many), the issue of revoking some users' access privileges arises but can be classically solved at the cost of key re-distribution and data re-encryption. In return for this computation cost, a new group member cannot decrypt the previous outsourced data with the new decrypting keys and a revoked user cannot decrypt any published data later with its keys, thanks to the *key refreshing* process.
- *Key management*– ID-based cryptography suffers naturally from the key escrow attack as a PKG needs to be defined. However, our solution mitigates this problem because each client acts as a PKG for his own data. That is, each client is responsible for generating the private keys needed for the decryption of his outsourced ciphered data. In addition, in any classical asymmetric cryptographic system, the distribution of public keys remains a burden as it requires the definition of a certification authority and the usage of certificates. However, in our proposal (e.g., sharing process), we avoid the need for public key distribution to other users thanks to the use of IBC.
- *Recovery*– The recovery of sensitive lost information remains an important security requirement in cloud storage environment, especially when CSPs are con-

sidered as untrusted agents. All the same, several public key cryptographic solutions require a system to carry a Trusted Computing Base (TCB) [21], in order to provide secure storage of secret keys and to post status updates. Otherwise, our contribution is responsive to any loss of data identifiers. Nonetheless, an exhaustive computing solution where clients' profiles are stored on cloud database with respective access rights on data. So that, any client requests to retrieve his stored data, relatively on a specific directory. Furthermore, the client may store data identifiers in cloud servers, based on his own indexing system.

Finally, our contribution is not restricted to any specific ID-based encryption scheme. So, instantiation of our proposal is given flexibility to implement appropriate ID-based encryption schemes. No change to the adopted ID-based schemes is made, hence, the security properties of the cryptographic primitives are well respected.

VII. Implementation results

The effort to evaluate the performance of our solution leads us to study the time performance of some well-known ID-based encryption schemes. Our tests are conducted in order to understand the execution cost of our proposal on real hardware. First, we implemented Boneh–Franklin [10], Boneh–Boyen [22] and Chen et al. [23] encryption algorithms using the Pairing-Based Cryptography (PBC) library [24]. Then, we evaluated their encryption and decryption times, of the same 10 kB block of random data for each IBE algorithm, using a symmetric pairing function (type A).

For our tests, we used 1000 samples in order to get our average times. In addition, we conducted our experiments on an Intel core 2 duo, started on *single mode*, where each core relies on 800 MHz clock frequency (CPU). The obtained results are summarized in Table II.

Security level (in bits)	80	112	128
Encryption time			
Boneh–Franklin	11.2	45.1	106.6
Boneh–Boyen	15.6	53.4	110.8
Chen et al.	5.9	19.8	41.4
Decryption time			
Boneh–Franklin	5.3	25.5	65.5
Boneh–Boyen	10.5	50.8	130.9
Chen et al.	5.3	25.4	65.4

TABLE II: IBE encryption and decryption duration (in ms).

Table II shows that the selection of algorithm and security level have great impact over time performances of IBE encryption and decryption operations. This is due to IBE algorithms integrating a varying number and type of operations in the group of elliptic curve (scalar and point multiplications) and pairing functions. For example, during the decryption phase, Boneh–Boyen algorithm took twice the time (10 ms) than Boneh–Franklin and Chen et al. algorithms put to end the decryption (5 ms). In fact, Boneh–Boyen relies on 2 pairing computations when the two other algorithms relies only on one pairing calculus.

Better IBE performances can be expected in the future with definition of new pairing functions like Beuchat et al. running in less than 1 ms [25]. However, IBE schemes remain slower than the classical AES encryption algorithm mostly used today by cloud storage providers. As a matter of fact, IBC should be considered as an interesting compromise between computation time and memory storage.

In addition, Table II shows that the consumed time for encryption or decryption increases, independently from the choice of the encryption algorithm, when we increase the level of security. The latter is recurrent concept in cryptography. It permits to evaluate the hardness of breaking an encryption or a signature algorithm. That is, the longer the level of security is, the harder the cryptanalysis of the algorithm becomes. For more details about the security level definition for ID-based encryption algorithms, please refer to Paterson's work [26].

The selected security level must be tightly adapted to the required security level for mitigating performance lowering. As such, the client can apply several IBC-PE and select one of them according to the sensitivity of his data. That is, for critical data, the client can choose IBC-PE with a higher security level (e.g. 128 bits). Consequently, the client's data encryption and decryption will last longer because the elliptic curve keys used for encryption and decryption will be around 256 bits long.

VIII. Conclusion

The growing need for secure cloud storage services and the attractive properties of ID-based cryptography lead us to combine them, thus, defining an innovative solution to the data outsourcing security issue.

Our solution is based on a specific usage of IBC. First, the cloud storage clients are assigned the IBC–PKG function. So, they can issue their own public elements, and can keep confidential their resulting IBC secret. Second, a per data key which is derived from a data identifier is used to encipher data.

Thanks to IBC properties, this contribution is shown to support data privacy and confidentiality, as it employs an original ID-based client side encryption approach. In addition, due to the lightweight ID-based public key computation process and contrary to the existing classical sharing schemes, our proposal does not require for the depositor to be connected, when the recipients want to retrieve the shared data. Moreover, our solution is also shown to be resistant to unauthorized access to data and to any data disclosure during the sharing process.

Finally, we believe that cloud data storage security is still full of challenges and of paramount importance, and many research problems remain to be identified.

Acknowledgements

This work is part of ODISEA project and is financially supported by the Conseil Regional d'Ile de France.

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009. [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

[2] D. Chappell, "Introducing the Windows azure platform," *October*, vol. 30, no. October, p. 2010, 2010. [Online]. Available: http://download.microsoft.com/download/C/0/2/C02C4D26-0472-4688AC13-199EA321135E/Introduce_Azure_Services_Platform_1_2.pdf

[3] Amazon, "Amazon simple storage service (amazon s3)." [Online]. Available: http://aws.amazon.com/s3/

[4] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proceedings of the 14th international conference on Financial cryptograpy and data security*, ser. FC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 136–149. [Online]. Available: http://dl.acm.org/citation.cfm?id=1894863.1894876

[5] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 85–90.

[6] D. Ratna, B. Rana, and S. Palash, "Pairing-based cryptographic protocols : A survey," 2004, http://eprint.iacr.org/.

[7] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53.

[8] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[9] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels, *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. New York, NY, USA: Cambridge University Press, 2005.

[10] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '01. London, UK, UK: Springer-Verlag, 2001, pp. 213–229. [Online]. Available: http://dl.acm.org/citation.cfm?id=646766.704155

[11] R. Sakai and M. Kasahara, "Id based cryptosystems with pairing on elliptic curve," Cryptology ePrint Archive, Report 2003/054, 2003. [Online]. Available: {http://eprint.iacr.org/}

[12] H. W. Lim and M. J. B. Robshaw, "On identity-based cryptography and grid computing," *Lecture Notes in Computer Science*, pp. 474–477, 2004. [Online]. Available: http://www.springerlink.com/content/ylj95gfgjxlb2131

[13] ——, "A dynamic key infrastructure for grid," in *Proceedings of the 2005 European conference on Advances in Grid Computing*, ser. EGC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 255–264.

[14] C. Schridde, T. Dörnemann, E. Juhnke, M. Smith, and B. Freisleben, "An identity-based security infrastructure for cloud environments," in *Proc. of IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS2010)*, 2010.

[15] H. W. Lim and K. G. Paterson, "Identity-based cryptography for grid security," *Int. J. Inf. Secur.*, vol. 10, no. 1, pp. 15–32, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10207-010-0116-z

[16] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-based authentication for cloud computing," in *Proceedings of the 1st International Conference on Cloud Computing*, ser. CloudCom '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 157–166. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10665-1_14

[17] T. Dierks and E. Rescorla, "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," Tech. Rep., Aug. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246

[18] T. Snia, "Cloud data management interface," *Representations*, pp. 1–173, 2010. [Online]. Available: http://www.snia.org/tech_activities/publicreview/CDMI_Spec_v08.pdf

[19] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of the Third international conference on Applied Cryptography and Network Security*, ser. ACNS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 442–455. [Online]. Available: http://dx.doi.org/10.1007/11496137_30

[20] A. Yaar, A. Perrig, and D. X. Song, "Fit: fast internet traceback," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*. IEEE, 2005, pp. 1395–1406.

[21] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *HOTCLOUD*. USENIX, 2009.

[22] D. Boneh and X. Boyen, "Efficient selective-id secure identity-based encryption without random oracles," pp. 223–238, 2004.

[23] L. Chen, Z. Cheng, J. Malone-Lee, and N. P. Smart, "Efficient id-kem based on the sakai-kasahara key construction," pp. 19–26, 2006.

[24] L. Ben, "On the implementation of pairing-based cryptosystems," 2007.

[25] J.-L. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya, "High-speed software implementation of the optimal ate pairing over barreto-naehrig curves," in *Proceedings of the 4th international conference on Pairing-based cryptography*, ser. Pairing'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 21–39.

[26] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," pp. 3113 – 3121, 2008, applications of Algebra to Cryptography. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X08000449